

-----Jun., 1991----- Issue #s-3-90----- TS Bulletin News Supplement Since '87----- 4 pages -----

© Jun., 1991 in the public domain. (Photocopy freely!)

This issue attacks the subject of learning a new computer language. It helps to know something about the philosophy of a computer language in order both to use it to its best advantage and even to learn it more rapidly. The article in this issue takes up a fundamental difference between BASIC and the structure conscious languages Pascal and C language. The use of 'doors' and 'walls' to control variable processing paths is a major stumbling block for the BASIC programmer wanting to get into the spirit of programming in Pascal or C language. The way that we take away that stumbling block in the article in this issue points out just how important a thing history is to understanding technical ideas. It is pretty hard to see where you are and find your way in new ideas when you have no idea of how we got this far. Ideas are put forth to solve technical problems and if you have no clue of the origin of the ideas as a solution to the problem or even what the problem was, how can you understand the new solution? Fortunately we BASIC programmers do not have to look further than our long and frustrating attempts to debug our last big BASIC program to see that variables can change anywhere in a BASIC program and finding out where they were wrongly changed or why they were not changed by a given block of code can take up an enormous amount of debugging time (if indeed we ever find out). Seeing the 'doors' and 'walls' put in Pascal and C language programs to keep the variable changes on fewer and better defined and so easier to track paths, will help the newcomer to Pascal or C language programming look at Pascal and C language source programs with a quicker understanding of what is going on. Or what is not but should be, when debugging a program of one's own. Happy thinking anyway!

This article will be followed by further exploring of how to avoid the pitfalls the amateur faces in switching from BASIC to Pascal or C language programming, in future newsletters too.

.....  
Recent News: Seattle TSUG N/L is also running articles on concepts of structure conscious languages like Pascal, Part 1 on Lists and Recursion, by Dale Fritz in Feb., 1991 issue, getting the topic off to an excellent start. This 28 page issue is available w. yearly subscription for US\$10 from SEATUG, c/o Dave Battershill, P.O.B. 64462, Tacoma, WA, USA 98464.... Toronto's Club N/L for Jan-Feb, 1991 is out and is thick too. Newsletters have their thick and thin issues but seeing some thick TS ones recently is most encouraging that the hobby will still be supported in the future. .... Look for some renewed vigour in TS / Sinclair North America User --- newsletters as a new editor is taking over, Mr. Donald Lambert, 1301 Kiblinger Pl, Auburn, IN USA 46706. He is seeking new articles, TS2068, LKDOS disk files most appreciated.... ISTUG, Indiana TSUG puts out a fine newsletter, 513 East Main St., Peru, IN, USA 46970. another excellent newsletter is Las Vegas 'Hacker', c/o Steve Sawchuk, 2345 Moorpark Way, Henderson, NV, USA 89015.... LIST is also active and has TS2068 tapes for sale as well as a newsletter, c/o 5 Peri Lane, Valley Stream, NY, USA 11581....

Computer Languages...-----Continued from page 3-----

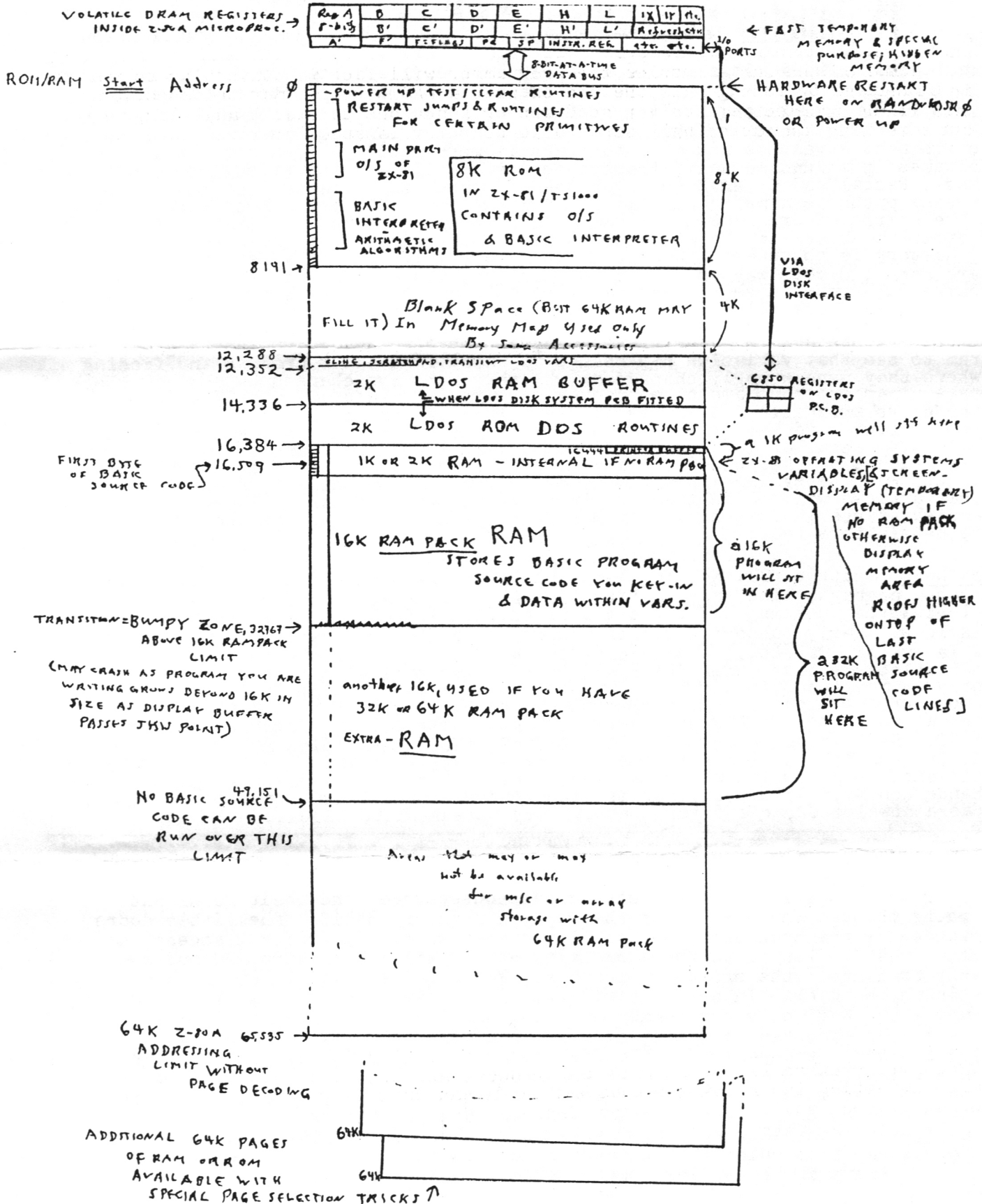
don't have to look all over for where they acquired their value, tracing long spaghetti-like strands of logical consequences and their resultant paths through the BASIC code the way you might in BASIC. The little doors formed by the brackets -cube( ) and even main( ), control access, so that what is let in to the subroutine can be strictly controlled and so the tracing of the program's actions. The { } are like walls\* that won't let anything else in that way either. BASIC having no such walls and doors, makes the beginning programmer entering the world of Pascal or C language feel like someone travelling from where people only lived under trees, and seeing doors and walls for the first time. Luckily it does not take an Einstein-like IQ to get used to these doors and walls, and the programmer new to Pascal and C language soon will forget that they were ever so strange and foreign. Since Pascal and C language are the amateur programmer's second choices beyond BASIC and machine language these days, FORTH and LOGO almost forgotten, it shouldn't be so hard on us now that we can see how they work.

-(Next in the series, more problems in adapting to Pascal & C solved)-

.....

\* in Pascal the walls are 'BEGIN' & 'END'.

Tourist's Map of ZX-81/TS1000 RAM/ROM Memory Addresses



There is a movement among serious hobbyists to move to programming using languages other than BASIC (or FORTH), C language probably being the most useful, and Pascal regaining its favour too.

One of the early puzzles the programmer will face in converting to working in C language or Pascal, is the use of what a BASIC programmer thinks of as a subroutine (called variously procedures or functions in Pascal and C language), but which has an exceedingly strange looking way of specifying variable use. Here it is:

```

cube(numbr);
{ int result
  result = numbr * numbr * numbr ;
  return(result);
}
1000 REM BASIC CUBE SUBROUTINE
1010 LET RESULT=0
1020 LET RESULT=NUMBR *NUMBR *NUMBR
1030 LET X=RESULT
1040 RETURN

```

This is an example of a subroutine to make the cube of a number in C language and is confusing to the BASIC programmer who is used to the code on the right above. To use the BASIC subroutine 1000, the programmer would, before calling, it put the number to be cubed in the variable NUMBR (with for example INPUT NUMBR). To print the result (the cube of NUMBR) after calling the subroutine with GOSUB 1000 you need only use: PRINT "THE CUBE="X and that is it. The variable X like the variable NUMBR exists in "the great variable table" and falls like manna from heaven onto any routine that needs it.

In Pascal and C language, the variables "numbr" and "result" are not kept indefinitely, so that the memory that they take up can be reused. They are immediately passed, on exit from the 'subroutine', to the procedure call calling them, so that the memory using them can be overwritten and thus reused by another variable. For this also to speed things up the memory used can be a small, stack memory so that variable value saving and retrieving can also be faster. In addition to speed and memory saving, (and any computer method that can do both at the same time can't be all bad), other tricks can be done with this method, unseen by the programmer, but of too advanced a nature to start discussing right here.

The main visible benefit to the programmer is that it makes structured programming easier and more obvious and keeps the programmer from losing track of variables the way they get lost in BASIC, sitting unused but not erased in the variable table, taking up room, slowing variable retrieval (as the table gets bloated in size) and causing confusion in the programmer's mind (and increasing the probability of errors through forgetting variables and their interaction).

Pascal and C language create a wall around the variables as far as the programmer is concerned with the curly brackets {}, so that variables can't automatically get into a subroutine or out of it, just by being named and used by the subroutine. The variable within the routine to take a value from outside is placed in the brackets, in the example, numbr, is placed in the brackets like this: cube(numbr). In the main part of the program, when the cube subroutine is called, the value to be taken from the main program and passed to the subroutine variable numbr, will be placed in that brackets, when the subroutine is called. For example:

C lang.-

```

{ printf("Enter number to be cubed n");
  scanf("%d",&saywhatcubed);
  x=cube(saywhatcubed);
  printf("The cube is %d n",x);
}

```

Printf is a print function from the C language function library and does the job of PRINT in BASIC. Scanf is another function that does the job of INPUT in BASIC. (The other little symbols and things including for carriage return used, are formatting instructions, that can be looked up in tables and simply have to be memorized, at least if you use them enough).

As soon as a variable or number is placed in the brackets, cube(5), cube(n), cube(fumblenum), or whatever, the value get passed through that door inside the walls that keep the variables pure and isolated in the function/procedure cube and the value is given to numbr, a variable that exists only inside those walls. The result exits through the backdoor, "return". This keeps tabs on variables so you (CANT)

One of the problems that faces the amateur electronics builder and hacker is getting specialized parts exact specifications. None of these parts is as elusive for the purchaser seeking budget prices as the quartz crystal or its ceramic crystal/piezo resonator equivalent.

The Sinclair ZX-81/Timex TS1000 uses a ceramic resonator rather than a quartz crystal for economy's sake. Probably a quartz crystal, if plugged in would work as well. The ZX-81/TS1000 also uses a three legged resonator although if you examine the circuit board carefully you will see that only two of the legs have traces leading anywhere, so that any 6.5 MHz two legged one will do as a substitute.

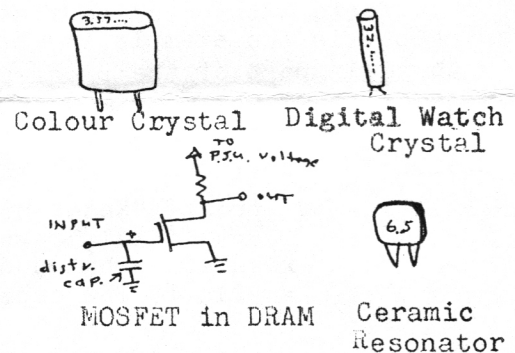
One problem though that you may get into in substituting or replacing quartz crystals is that there are two kinds, the series resonant and the parallel resonant (for each frequency) and one will not often work properly in place of the other kind. Reference to a text book on this will offer an explanation that we can't go on with here except to say that it is similar to series and parallel resonant LC, tuned circuits in radio electronics that even if tuned to the same frequency differ.

The one quartz crystal that you can buy everywhere is the 3.57 (and a bunch more decimal place figures) MHz colour crystal for t.v. sets. Radio Shack usually stocks them and of course you can get them out of old, junked t.v.'s (but in t.v.'s some use the parallel and some the incompatible series type). A local user group member, unwilling to spend on an expensive quartz crystal for a speech synthesis project using an i.c. from Radio Shack, substituted a 3.57 MHz colour t.v. crystal with only a little voice change resulting in the synthesized speech.

If he had known that most quartz clocks use a crystal, typically 4.19 MHz, that can be salvaged from them when they stop working, he could have obtained a crystal even closer to the target frequency.

There are only a few projects that the writer has seen in the magazines for them, but one could envision many possibilities for the use of the little 32.268 KHz, low frequency, crystals in digital watches. Using them in microprocessors might not be possible since most microprocessors have a minimum operating frequency. The Z-80, for example, has a minimum clock frequency of about one-third its rated frequency and one would expect other microprocessors to be similar. This is due to the Z-80 using DRAM for its register memory, so that it must be refreshed (and it is refreshed at the pace of the clock signals) before the little distributed capacitances that are holding a charge to the gate of the little MOSFET transistors, to preserve the bit that it is storing, run out (lose their charge)\*. However, the writer has not scoured all the spec books for microprocessors in search of one that does not require this, and so would work off a 32 KHz clock signal.

It is interesting though that one can get such a fascinating piece of high technology as a quartz crystal for free, simply for the trouble of opening up a junked digital clock or watch and unsoldering it with a small pencil soldering iron.



- \* Too slow a clock would not recharge these little capacitances in time before their charge would run out, these internal registers connected to internal refresh circuitry, not user accessible. Funny to use dynamic RAM for internal registers, but that's what they did, and likely many other microprocessor design teams too.

Note: Active Components parts stores are one of the few to carry a range of ceramic resonators (and they also carry crystals).